Predicting Physics Observables by Analysing LHC like Data

Kain Shivendrasingh
Roll No.: 24B1817

Mentor: Deependra Sharma

WiDS, Analytics Club, IIT Bombay

February 1, 2026

Abstract

This document serves as a structured log and technical report for the project *Predicting Physics Observables by Analysing LHC like Data*. The report documents the step-by-step exploration of ROOT files, understanding of data structures, and preliminary analysis techniques used in high-energy physics workflows.

Contents

1 Introduction

High-energy physics experiments such as those conducted at the Large Hadron Collider (LHC) generate extremely large and structured datasets. These datasets are commonly stored using the ROOT framework, which provides efficient data storage, access, and visualization capabilities.

The objective of this project is to develop a systematic understanding of ROOT files and their internal structure as a first step toward predicting physics observables from LHC data.

2 Understanding ROOT

Exploration of AnalysisResults67.root

This section documents the systematic exploration of the ROOT file `AnalysisResults67.root`. The objective was to understand the internal structure of the file, identify the types of stored objects, and verify whether the data is stored in event-wise tabular form or as aggregated analysis outputs.

2.1 Overview of the ROOT File Structure

The file was opened using the ROOT framework and its top-level contents were listed. The file does not contain any objects directly at the top level; instead, it is organized into multiple task-based directories.

```
TFile *f = TFile::Open("AnalysisResults67.root");
f->ls();
```

The output revealed the following top-level directories:

- `track-propagation`
- `tof-signal`
- `event-selection-task`
- `bc-selection-task`
- `ft0-corrected-table`
- `centrality-table`
- `hf-candidate-creator-dstar`
- `hf-candidate-creator-dstar-expressions`
- `hf-candidate-selector-dstar-to-d0-pi`
- `hf-task-dstar-to-d0-pi`

This directory-based organization indicates that the file corresponds to an *analysis output* rather than raw or reduced event-level data.

2.2 Nested Directory Exploration

Each directory was explored individually using the `ls()` method. As a representative example, the contents of the `track-propagation` directory are shown below:

```
f->Get("track-propagation")->ls();
```

This directory contains the following objects:

- `hDCAxyVsPtRec` (TH2F)
- `hDCAxyVsPtMC` (TH2F)
- `hDCAzVsPtRec` (TH2F)
- `hDCAzVsPtMC` (TH2F)
- `trackTunedTracks` (TH1D)

No `TTree` objects were found in this directory.

2.3 Printing the Data Structure Using `uproot`

To programmatically inspect the data structure in Python, the `uproot` library was used. Listing all keys in the file returned both directories and their nested objects:

```
import uproot
file = uproot.open("AnalysisResults67.root")
file.keys()
```

Since `uproot` returns only object paths, the ROOT class of each object was determined explicitly using the `classname` attribute.

For the `track-propagation` directory, the following code was used:

```
for key in file["track-propagation"].keys():
    obj = file[f"track-propagation/{key}"]
    print(f"{key} -> {obj.classname}")
```

The output confirms the histogram-only nature of the directory:

```
hDCAxyVsPtRec;1      -> TH2F
hDCAxyVsPtMC;1       -> TH2F
hDCAzVsPtRec;1       -> TH2F
hDCAzVsPtMC;1        -> TH2F
trackTunedTracks;1  -> TH1D
```

2.4 Detailed Histogram Inspection

A representative two-dimensional histogram, `track-propagation/hDCAxyVsPtRec`, was inspected in detail using `uproot`.

```
h = file["track-propagation/hDCAxyVsPtRec"]

print("Class:", h.classname)
print("Axes:", h.axes)
print("Number of bins:", [len(ax.edges()) - 1 for ax in h.axes])
```

The output is:

```
Class: TH2F
Axes: (<TAxis>, <TAxis>)
Number of bins: [600, 55]
```

This confirms that the object is a two-dimensional floating-point histogram with 600 bins along the first axis and 55 bins along the second axis.

2.5 Absence of TTrees and Pandas-Based Analysis

All directories in the file were systematically explored using both ROOT and `uproot`. No `TTree` objects were found anywhere in the file. Instead, the file exclusively contains histogram objects (`TH1` and `TH2`) organized in nested, task-based directories.

Since pandas is designed for event-wise tabular data (typically stored in `TTree` objects), pandas-based exploration was not applicable for this file. The data represents aggregated analysis outputs rather than per-event records.

2.6 Key Observations

- The ROOT file is structured as a collection of task-based directories.
- All data is stored in the form of histograms (`TH1F`, `TH1D`, `TH2F`).
- No event-wise `TTree` objects are present.
- Python-based inspection using `uproot` provides a clean and programmatic way to classify ROOT objects and extract structural information.

Exploration of `Prompt_DstarToD0Pi.root`

This section presents a detailed and conceptual explanation of the structure and contents of the ROOT file `Prompt_DstarToD0Pi.root`. The objective is not only to list the contents of the file, but also to understand how ROOT stores event-wise data and how such data is interpreted in the context of physics and machine learning analyses.

2.7 Top-Level Structure of the ROOT File

The file was opened using the ROOT framework and its contents were listed using the command:

```
TFile *f = TFile::Open("Prompt_DstarToD0Pi.root");
f->ls();
```

The output of this command revealed a single object stored at the top level:

- `treeMLDstar` (TTree)

In ROOT, a `TTree` represents a collection of event-wise data. The absence of directories or histogram objects indicates that this file is not an analysis output or summary file, but rather a dataset intended for further processing, such as statistical analysis or machine learning.

2.8 Concept of a TTree in ROOT

A `TTree` can be understood as a table where each row corresponds to one event (or candidate) and each column corresponds to a physical or derived variable. In this file, each entry of the tree corresponds to a reconstructed $D^* \to D^0\pi$ candidate.

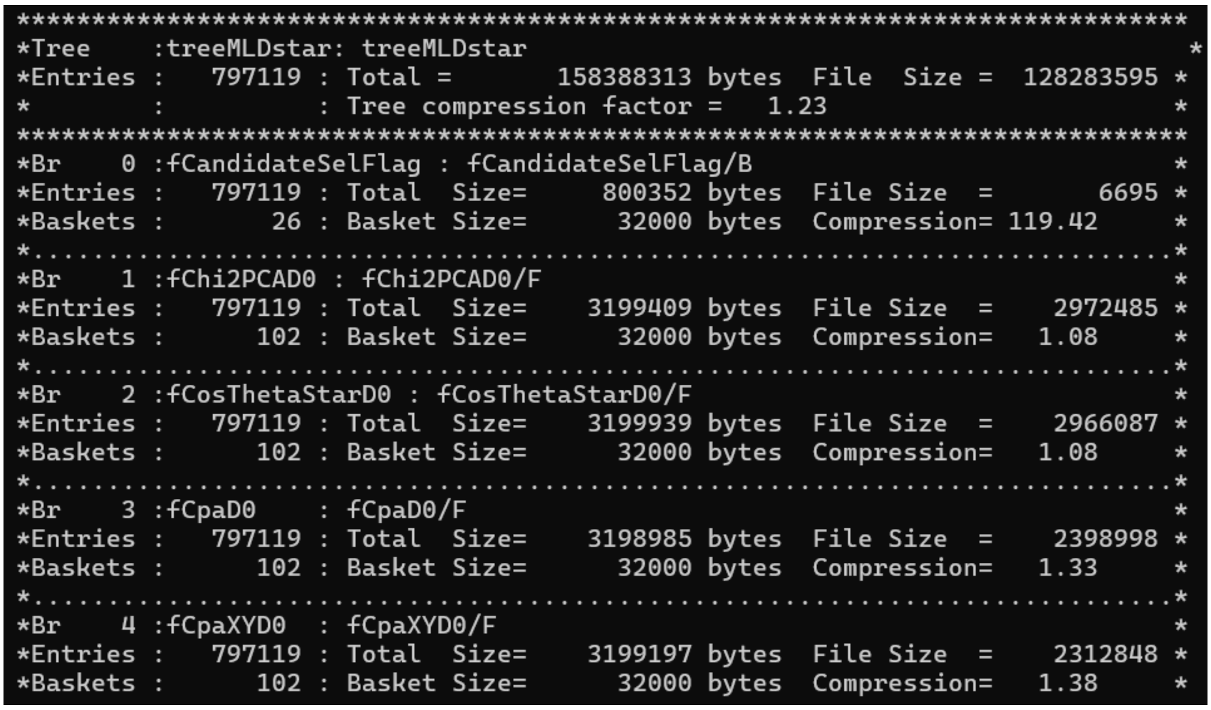
An important feature of ROOT TTrees is that they use *columnar storage*. This means that each variable (branch) is stored independently as a contiguous array of values, rather than storing complete rows together. Such a design allows efficient access to selected variables and enables strong data compression, which is crucial for large datasets encountered in LHC experiments.

2.9 Inspecting the Tree Structure

The structure of the tree was inspected using:

```
TTree *t = (TTree*)f->Get("treeMLDstar");
t->Print();
```

The header of the output provides global information about the dataset:



```
******************************************************************************
*Tree    :treeMLDstar: treeMLDstar                                            *
*Entries :   797119 : Total =       158388313 bytes  File  Size =  128283595 *
*        :          : Tree compression factor =   1.23                       *
******************************************************************************
*Br    0 :fCandidateSelFlag : fCandidateSelFlag/B                            *
*Entries :   797119 : Total  Size=     800352 bytes  File Size  =       6695 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression= 119.42     *
*............................................................................*
*Br    1 :fChi2PCAD0 : fChi2PCAD0/F                                          *
*Entries :   797119 : Total  Size=    3199409 bytes  File Size  =    2972485 *
*Baskets :      102 : Basket Size=      32000 bytes  Compression=   1.08     *
*............................................................................*
*Br    2 :fCosThetaStarD0 : fCosThetaStarD0/F                                *
*Entries :   797119 : Total  Size=    3199939 bytes  File Size  =    2966087 *
*Baskets :      102 : Basket Size=      32000 bytes  Compression=   1.08     *
*............................................................................*
*Br    3 :fCpaD0    : fCpaD0/F                                               *
*Entries :   797119 : Total  Size=    3198985 bytes  File Size  =    2398998 *
*Baskets :      102 : Basket Size=      32000 bytes  Compression=   1.33     *
*............................................................................*
*Br    4 :fCpaXYD0  : fCpaXYD0/F                                             *
*Entries :   797119 : Total  Size=    3199197 bytes  File Size  =    2312848 *
*Baskets :      102 : Basket Size=      32000 bytes  Compression=   1.38     *
```

Figure 1

- **Tree name:** `treeMLDstar`
- **Number of entries:** 797,119

The number of entries corresponds to the total number of candidates stored in the dataset. Each entry represents one independent event or candidate, and all branches have the same number of entries, ensuring a consistent tabular structure.

2.10 Understanding Branches and Data Types

Following the header, the output lists individual branches. Each branch corresponds to one variable stored in the dataset. A typical branch entry has the form:

```
*Br   42 :fPt : fPt/F
```

This information can be interpreted as follows:

- `fPt` is the name of the branch.
- The suffix `/F` indicates that the data type is `Float_t`, meaning a single-precision floating-point number.

Similarly, branches ending with `/B` correspond to boolean values (`Bool_t`), typically used as selection flags or labels.

Importantly, all branches in this tree store scalar values. There are no array or vector branches, which implies that the tree is flat and each entry contains exactly one value per branch. This structure is ideal for conversion to tabular formats such as pandas DataFrames.

2.11 Columnar Nature of the Data

From a data-organization perspective, the tree can be viewed as a set of columns:

- Kinematic variables such as `fPt`, `fEta`, `fPhi`
- Topological variables such as `fDecayLengthD0` and `fImpactParameter`
- Particle identification variables such as `fNSigTpc` and `fNSigTof`
- Selection and matching flags stored as boolean branches

Each of these branches is stored independently and can be accessed without loading the entire dataset, which is a key advantage of ROOT's columnar design.

2.12 Python-Based Inspection Using `uproot` and `pandas`

The tree was also accessed using the Python library `uproot`, which allows ROOT files to be read without relying on the ROOT C++ framework:

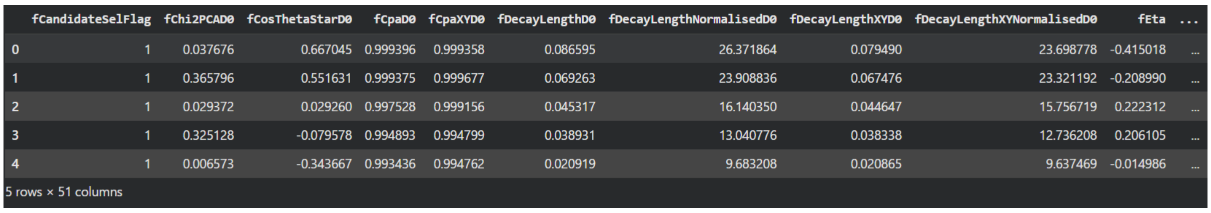
```
import uproot
file = uproot.open("Prompt_DstarToD0Pi.root")
tree = file["treeMLDstar"]
```

The tree was converted into a pandas DataFrame using:

```
df = tree.arrays(library="pd")
df.shape
```

The resulting DataFrame has shape:

```
(797119, 51)
```



	fCandidateSelFlag	fChi2PCAD0	fCosThetaStarD0	fCpaD0	fCpaXYD0	fDecayLengthD0	fDecayLengthNormalisedD0	fDecayLengthXYD0	fDecayLengthXYNormalisedD0	fEta	...
0	1	0.037676	0.667045	0.999396	0.999358	0.086595	26.371864	0.079490	23.698778	-0.415018	...
1	1	0.365796	0.551631	0.999375	0.999677	0.069263	23.908836	0.067476	23.321192	-0.208990	...
2	1	0.029372	0.029260	0.997528	0.999156	0.045317	16.140350	0.044647	15.756719	0.222312	...
3	1	0.325128	-0.079578	0.994893	0.994799	0.038931	13.040776	0.038338	12.736208	0.206105	...
4	1	0.006573	-0.343667	0.993436	0.994762	0.020919	9.683208	0.020865	9.637469	-0.014986	...

5 rows × 51 columns

Figure 2

This confirms that the dataset contains 797,119 rows (entries) and 51 columns (branches). A preview of the column names and sample rows was inspected visually and included in the report as a screenshot for clarity.

2.13 Summary of Observations

- The file contains a single `TTree` storing event-wise data.
- The tree is flat and columnar, with scalar branches only.
- The structure is well suited for statistical analysis and machine learning workflows.
- The ROOT and Python-based views of the data are consistent with each other.

3 Fundamental Particle Physics Background

3.1 Summary of Relativistic Kinematics

Relativistic kinematics provides the correct framework for describing energy, momentum, and particle production in high-energy and nuclear collisions, where particle velocities are comparable to the speed of light. Classical mechanics fails in this regime, necessitating the use of Lorentz-invariant quantities and four-vector formalism.

3.1.1 Motivation for High Energies

The spatial resolution of a probe is limited by its de Broglie wavelength,

$$\lambda = \frac{h}{p}.$$

To probe sub-nuclear length scales (~ 1 fm), beams with momenta of order GeV are required. High beam energies also enable particle production via mass–energy equivalence,

$$E = mc^2.$$

3.1.2 Special Relativity and Lorentz Invariance

In relativistic dynamics, space and time combine into a four-dimensional space–time. Physical laws must be invariant under Lorentz transformations. The invariant space–time interval is

$$ds^2 = c^2dt^2 - dx^2 - dy^2 - dz^2,$$

which has the same value in all inertial frames.

3.1.3 Four-Vectors and Energy–Momentum Relation

Particle kinematics is described using four-vectors. The energy–momentum four-vector is

$$p^\mu = \left(\frac{E}{c}, \vec{p}\right),$$

with Lorentz-invariant magnitude

$$p^\mu p_\mu = m^2c^2.$$

This leads to the fundamental relativistic relation

$$E^2 = p^2c^2 + m^2c^4.$$

3.1.4 Natural Units

In high-energy physics, natural units are used by setting

$$\hbar = c = 1.$$

In this system, energy, momentum, and mass are all expressed in units of GeV, simplifying relativistic equations to

$$E^2 = p^2 + m^2.$$

3.1.5 Fixed-Target versus Collider Experiments

In fixed-target experiments, a high-energy projectile strikes a stationary target. The available center-of-mass energy scales as

$$E_{\text{CM}} \propto \sqrt{E_{\text{beam}}}.$$

In collider experiments, two beams collide head-on with equal and opposite momenta, yielding

$$E_{\text{CM}} \propto E_{\text{beam}}.$$

Thus, colliders are far more efficient for producing high-mass particles and accessing new physics.

3.1.6 Rapidity and Lorentz Boosts

Rapidity is introduced to simplify the description of relativistic motion along the beam axis. It is defined as

$$y = \frac{1}{2} \ln \frac{E + p_z}{E - p_z}.$$

Unlike velocity, rapidity is additive under Lorentz boosts, making it a natural variable for collider physics. Under a boost,

$$y \to y + y_{\text{boost}}.$$

3.1.7 Transverse Momentum and Transverse Mass

The momentum of a particle is decomposed into longitudinal and transverse components. The transverse momentum,

$$p_T = \sqrt{p_x^2 + p_y^2},$$

is invariant under boosts along the beam direction. The transverse mass is defined as

$$m_T = \sqrt{m^2 + p_T^2}.$$

3.1.8 Pseudorapidity

For highly relativistic particles ($E \gg m$), rapidity is approximated by pseudorapidity,

$$\eta = -\ln \tan \frac{\theta}{2},$$

which depends only on the polar angle θ. Detector geometries are therefore commonly designed in (η, ϕ) coordinates.

3.1.9 Light-Cone Variables

Light-cone momentum components are defined as

$$p^{\pm} = E \pm p_z.$$

These variables scale simply under Lorentz boosts and are useful in describing longitudinal momentum fractions, such as the Feynman scaling variable

$$x_F = \frac{2p_L}{\sqrt{s}}.$$

3.1.10 Invariant Yields and Particle Spectra

The Lorentz-invariant single-particle yield is expressed as

$$E\frac{d^3N}{dp^3}.$$

Experimentally, particle production is studied using transverse momentum (p_T) or transverse mass (m_T) spectra. The low-p_T region reflects thermal behavior, while the high-p_T region is dominated by hard QCD processes.

3.1.11 Collective Flow and Effective Temperature

In heavy-ion collisions, collective expansion (radial flow) modifies particle spectra. The effective temperature extracted from transverse momentum distributions is

$$T_{\text{eff}} = T_{\text{th}} + \alpha m v_{\text{flow}}^2,$$

where T_{th} is the thermal temperature and v_{flow} characterizes collective motion.

3.1.12 Physical Significance

Relativistic kinematics underpins the interpretation of high-energy collision data. Concepts such as invariant mass, center-of-mass energy, rapidity, and transverse momentum enable frame-independent comparisons between experiments and provide direct access to the dynamics of particle production and the properties of strongly interacting matter.

3.2 Elementary and Composite Particles

In modern particle physics, all known particles are classified into two broad categories: elementary particles and composite particles. Elementary particles are point-like objects with no known internal structure. These include fermions, such as quarks and leptons, and bosons, such as photons, gluons, and the Higgs boson. Composite particles are bound states of elementary particles held together by fundamental interactions.

Quarks are elementary fermions that carry fractional electric charge and participate in the strong interaction. A crucial property of the strong interaction is confinement: quarks and gluons are never observed as free particles in nature. Instead, they are permanently confined inside composite particles known as hadrons.

3.3 Hadrons: Mesons and Baryons

Hadrons are composite particles formed from quarks bound by the strong interaction, which is described by Quantum Chromodynamics (QCD). Hadrons are divided into two categories. Baryons consist of three quarks, such as the proton (uud) and neutron (udd). Mesons consist of one quark and one antiquark.

The particles studied in this work, namely pions, D^0, D^{*+}, and B mesons, all belong to the meson family.

3.4 Pions and Electric Charge

Pions (π mesons) are the lightest mesons and appear in three charge states: π^+, π^-, and π^0. Their electric charge is determined by their quark content. For example, π^+ is composed of an up quark and an anti-down quark, resulting in a net charge of +1. The existence of multiple charge states reflects the underlying quark structure and conservation of electric charge in strong interactions.

4 Heavy Quarks and Heavy-Flavor Mesons

4.1 Charm and Bottom Quarks

Charm (c) and bottom (b) quarks are referred to as heavy quarks due to their large masses compared to the QCD scale Λ_{QCD}. Heavy quarks are predominantly produced in hard partonic scatterings during the initial stage of high-energy proton–proton collisions. Because of their early production time, heavy quarks serve as clean probes of the strong interaction.

4.2 D and B Mesons

A D^0 meson is composed of a charm quark and an up antiquark ($c\bar{u}$), while B mesons contain a bottom quark paired with a light antiquark. Bottom mesons have longer lifetimes than charm mesons due to the weaker decay of the bottom quark. This difference in lifetime leads to distinct decay topologies that can be experimentally resolved.

4.3 Prompt and Non-Prompt Production

Heavy-flavor mesons are categorized according to their production mechanism. Prompt D mesons are produced directly from the hadronization of charm quarks created in the initial proton–proton collision. Non-prompt D mesons originate from the decay of longer-lived B mesons. Although both originate from the same collision, their decay vertices are spatially separated due to the longer lifetime of the parent B meson.

5 Strong Interaction: QCD, pQCD, and Deconfinement

5.1 Quantum Chromodynamics

Quantum Chromodynamics is the quantum field theory describing the strong interaction between quarks and gluons. A defining feature of QCD is confinement: the force between quarks increases with separation, preventing the observation of isolated quarks. As a result, quarks are always bound into hadrons under normal conditions.

5.2 Perturbative QCD

At sufficiently high energies, the strong coupling constant becomes small, allowing the use of perturbation theory. This regime is known as perturbative QCD (pQCD). Heavy-quark production in high-energy collisions is well described within pQCD, making charm and bottom quarks valuable probes for testing the theory.

5.3 Quark–Gluon Plasma

At extremely high temperatures and energy densities, such as those achieved in heavy-ion collisions, hadronic matter undergoes a phase transition into a deconfined state known as the quark–gluon plasma. In this state, quarks and gluons are no longer confined within individual hadrons but propagate freely over nuclear volumes. Heavy quarks produced before the formation of the plasma traverse this medium and lose energy, providing information about the plasma properties.

6 Experimental Reconstruction of D Mesons

6.1 Decay Channels

The D^{*+} meson is reconstructed via the decay chain

$$D^{*+} \to D^0 \pi^+_{\text{soft}}, \quad D^0 \to K^- \pi^+.$$

The small mass difference between the D^{*+} and the D^0 results in a low-momentum pion, referred to as a soft pion. This characteristic provides a powerful handle for suppressing combinatorial background.

6.2 Invariant Mass Reconstruction

The invariant mass of a parent particle is reconstructed from the energies and momenta of its decay products according to

$$m^2 = (E_1 + E_2)^2 - (\vec{p}_1 + \vec{p}_2)^2.$$

For genuine decays, repeated reconstruction across many events results in a peak at the true mass of the parent particle. Random combinations of tracks form a smooth background underneath the peak.

6.3 Transverse Momentum

The transverse momentum p_T is defined as

$$p_T = \sqrt{p_x^2 + p_y^2}.$$

It represents the component of momentum perpendicular to the beam axis and is invariant under boosts along the beam direction. Transverse momentum is a key observable in collider physics.

7 Decay Topology and Machine Learning

7.1 Decay Geometry

Particles with finite lifetimes decay at measurable distances from the primary vertex. The decay geometry is quantified using variables such as decay length, impact parameter, and pointing angle. The pointing angle θ_p is defined as

$$\cos\theta_p = \frac{\vec{p}_D \cdot \vec{L}}{|\vec{p}_D||\vec{L}|},$$

where $\vec{L}$ is the vector connecting the primary and secondary vertices. Values close to unity indicate a decay consistent with the reconstructed flight direction.

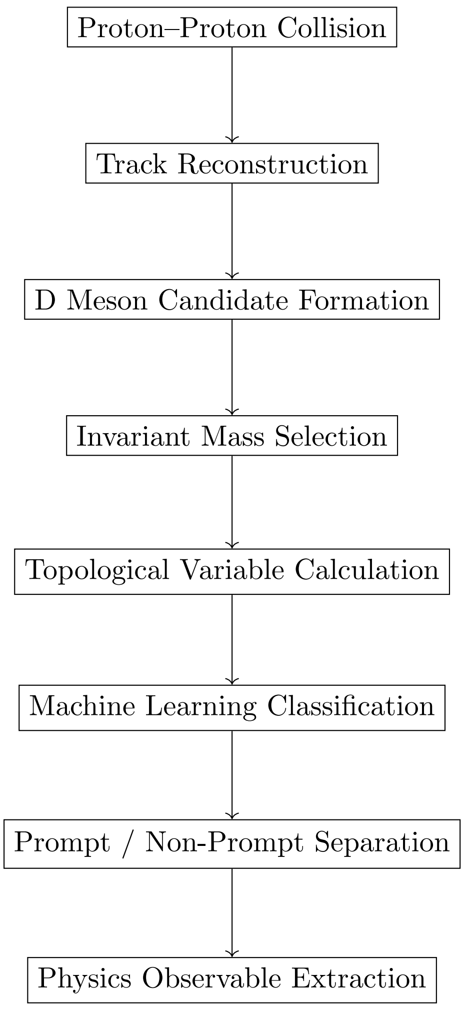
7.2 Machine Learning Classification

Machine learning techniques are employed to separate background candidates from real D mesons and to distinguish between prompt and non-prompt production. Only topological variables related to decay geometry are used as input features. Physics observables such as invariant mass and transverse momentum are explicitly excluded from the training to avoid biasing the extracted yields.

The classifier outputs probabilities for each candidate corresponding to background, prompt, and non-prompt classes. These probabilities are later used in the statistical extraction of physics observables.

8 Analysis Flow Summary

8.1 Analysis Workflow


Proton–Proton Collision
Track Reconstruction
D Meson Candidate Formation
Invariant Mass Selection
Topological Variable Calculation
Machine Learning Classification
Prompt / Non-Prompt Separation
Physics Observable Extraction

9 Dataset Structure and Variable Inspection

9.1 Overview of the Input ROOT Files

The analysis is based on three ROOT files corresponding to different physical classes of reconstructed D^* candidates:

- `Prompt_DstarToD0Pi.root`: D^* mesons produced promptly at the primary interaction vertex.

- `Nonprompt_DstarToD0Pi.root`: D^* mesons originating from weak decays of beauty hadrons.
- `Bkg_DstarToD0Pi.root`: combinatorial background candidates.

All three files share an identical internal structure and contain a single `TTree` named `treeMLDstar`. Each entry of the tree corresponds to one reconstructed D^* candidate formed via the decay channel

$$D^{*+} \to D^0\pi^+, \quad D^0 \to K^-\pi^+,$$

and its charge-conjugate mode. The datasets contain a large number of candidates, with approximately 1.1×10^7 entries in the prompt and background samples and $\sim 9.9 \times 10^5$ entries in the non-prompt sample.

It is important to emphasize that a tree entry does *not* correspond to a single collision event or run, but rather to a reconstructed D^* candidate. Multiple candidates may originate from the same event.

9.2 Branch Consistency and Data Validation

A detailed inspection of the tree structure using the ROOT framework confirms that all three files contain identical branches with consistent data types. This structural uniformity is essential for supervised machine-learning applications, as it ensures that the same set of input features is available for all classes.

The inspection step serves as a validation of data integrity and defines the complete feature space available for subsequent analysis stages.

9.3 Classification of Variables

The branches in `treeMLDstar` can be grouped into three broad categories: topological variables, kinematic variables, and particle-identification (PID) variables.

9.3.1 Topological Variables

The following topological observables are used as input features for the machine-learning classifier:

- `fCpaD0`: cosine of the pointing angle between the reconstructed D^0 momentum and the line connecting the primary and secondary vertices.
- `fCpaXYD0`: transverse-plane component of the D^0 pointing angle.
- `fDecayLengthXYD0`: transverse decay length of the D^0 meson.
- `fImpactParameterProductD0`: product of the impact parameters of the D^0 decay prongs.
- `fImpParamSoftPi`: impact parameter of the soft pion from the D^* decay.
- `fMaxNormalisedDeltaIPD0`: maximum normalized difference in impact parameters of the D^0 decay tracks.

These variables encode the displaced-vertex topology characteristic of heavy-flavor decays and provide strong discrimination power between prompt, non-prompt, and background candidates. Importantly, they are largely independent of the absolute momentum scale, which motivates their use as the primary training features.

9.3.2 Kinematic Variables

In addition to the topological observables, several kinematic quantities are available in the dataset:

- `fPt`: transverse momentum of the reconstructed D^* candidate.
- `fM`: invariant mass of the D^* candidate.
- `fMD0`: invariant mass of the reconstructed D^0 meson.
- `fInvDeltaMass`: mass difference $\Delta M = M(D^*) - M(D^0)$.

These variables are *not* used as training inputs, but are instead employed for data selection, validation, and physics interpretation, such as defining transverse-momentum intervals and verifying signal mass peaks.

9.3.3 Particle Identification Variables

The tree also contains a comprehensive set of PID-related observables based on TPC and TOF information (e.g. `fNSigTpc`, `fNSigTof`). These variables are intentionally excluded from the present training stage and are reserved for future extensions of the analysis.

9.4 Purpose of the Inspection Stage

The ROOT-based inspection performed in this stage serves three essential purposes:

1. Verification of dataset consistency across different physical classes.
2. Identification and categorization of variables relevant for machine-learning training and physics validation.
3. Establishment of a reliable and reproducible data foundation before transitioning to Python-based analysis using `uproot`.

Following this validation step, all subsequent analysis tasks, including plotting, feature engineering, and model training, are performed using Python-based tools.

10 Transverse Momentum Binning

10.1 Definition of Transverse Momentum

The transverse momentum p_T of a particle is defined as the component of its momentum perpendicular to the beam direction,

$$p_T = \sqrt{p_x^2 + p_y^2},$$

where the z-axis is aligned with the LHC beam. In this analysis, the variable `fPt` corresponds to the transverse momentum of the reconstructed D^* meson candidate, rather than that of its individual decay products.

The use of transverse momentum is particularly advantageous in hadron collider experiments, as it is invariant under longitudinal Lorentz boosts and is less sensitive to the unknown momentum of the initial-state partons.

10.2 Motivation for pT-Differential Analysis

Both the detector performance and the underlying physics of heavy-flavor production exhibit a strong dependence on p_T. At low transverse momentum, decay vertices are less displaced and topological variables show reduced separation power. At higher p_T, heavy-flavor decay topologies become more pronounced due to increased Lorentz boost, leading to improved discrimination between prompt and non-prompt candidates.

From a machine-learning perspective, the distributions of topological observables are strongly p_T dependent. Training a single classifier over a wide p_T range can therefore introduce biases and degrade performance. For this reason, modern heavy-flavor analyses perform classification in narrow p_T intervals.

10.3 Choice of pT Ranges

The analysis is performed using transverse-momentum intervals defined by the bin edges

$$[1,\ 1.5,\ 2,\ 3,\ 4,\ 6,\ 8,\ 10,\ 12,\ 16,\ 24]\ \mathrm{GeV}/c.$$

Each interval corresponds to an independent subset of reconstructed D^* candidates selected according to their `fPt` value.

For the final detailed study presented in this report, a single representative p_T interval is selected. This approach allows a focused investigation of the classifier performance while maintaining sufficient statistical precision and avoiding unnecessary complexity.

11 Transverse Momentum Binning

11.1 Definition of Transverse Momentum

The transverse momentum p_T of a particle is defined as the component of its momentum perpendicular to the beam direction,

$$p_T = \sqrt{p_x^2 + p_y^2},$$

where the z-axis is aligned with the LHC beam. In this analysis, the variable `fPt` corresponds to the transverse momentum of the *reconstructed D^* meson candidate*, and not to that of the individual decay tracks.

Each entry in the input dataset represents a reconstructed D^* candidate. Consequently, applying a selection on p_T reduces the number of reconstructed candidates under consideration, rather than the number of collision events. A single event may contain zero, one, or multiple D^* candidates.

The use of transverse momentum is particularly advantageous in hadron-collider experiments, as it is invariant under Lorentz boosts along the beam direction and is less sensitive to the unknown longitudinal momenta of the initial-state partons.

11.2 Physical Motivation for pT-Differential Analysis

The production rate and decay topology of heavy-flavor hadrons exhibit a strong dependence on transverse momentum. The p_T spectrum of produced particles falls rapidly with increasing p_T, reflecting the underlying dynamics of parton–parton interactions. Low-p_T particles can be produced through soft scatterings, which are abundant, whereas high-p_T particles require rare, hard scatterings involving partons carrying a large fraction of the proton momentum.

As a result, the particle yield approximately follows a power-law behavior,

$$\frac{dN}{dp_T} \propto p_T^{-n},$$

with $n \sim$ 4–8, implying that particles with higher transverse momentum are produced with significantly lower probability.

Despite their reduced yield, higher-p_T D^* mesons exhibit more pronounced decay topologies. Due to the increased Lorentz boost, heavy-flavor hadrons at higher p_T travel a longer distance before decaying,

$$L \sim \beta\gamma c\tau,$$

leading to larger decay lengths, improved vertex separation, and better-defined pointing angles. Consequently, topological observables provide stronger discrimination power between prompt, non-prompt, and background candidates at higher p_T.

11.3 Motivation from a Machine-Learning Perspective

From a machine-learning standpoint, the distributions of topological variables such as decay length, impact parameter, and pointing angle are strongly p_T dependent. Training a single classifier over a wide p_T range would allow the model to exploit kinematic correlations rather than genuine topological differences, leading to biased learning and reduced generalization capability.

To avoid this effect, the classification task is performed in narrow p_T intervals. Within a fixed p_T range, the model is forced to learn differences arising from the production mechanism and decay topology, rather than from trivial kinematic scaling.

11.4 Choice of pT Intervals and Selected Analysis Range

The transverse-momentum intervals considered in this analysis are defined by the bin edges

$$[1, 1.5, 2, 3, 4, 6, 8, 10, 12, 16, 24] \text{ GeV}/c.$$

Each interval corresponds to a subset of reconstructed D^* candidates selected according to their `fPt` value.

For the detailed analysis presented in this report, the $6 < p_T < 8$ GeV/c interval is selected. This range provides a favorable balance between statistical precision and topological separation power, while avoiding regions dominated by low-p_T resolution effects or limited high-p_T statistics. The selected interval therefore serves as a representative case for demonstrating the classification methodology and performance.

12 Exploratory Analysis of Topological and Kinematic Variables

Before training any multivariate classifier, it is essential to validate that the chosen input variables possess genuine physical discriminating power between different classes of reconstructed candidates. In this section, we perform a detailed exploratory data analysis (EDA) of the topological and kinematic variables used to separate prompt, non-prompt, and background $\mathrm{D}^{*+} \to \mathrm{D}^0\pi^+$ candidates.

12.1 Nature of the Data: Candidates vs Events

The ROOT files used in this analysis store reconstructed *candidates*, not collision events. Each entry in the TTree `treeMLDstar` corresponds to a single reconstructed D^* candidate formed from a D^0 and a soft pion track. Multiple candidates may originate from the same collision event, and the event structure itself is not explicitly used in the analysis.

As a result, all distributions presented in this section represent *candidate-level* quantities, and the vertical axis of all histograms corresponds to the number (or normalized density) of reconstructed candidates.

12.2 Prompt, Non-prompt, and Background Classes

Three distinct classes of candidates are considered:

- **Prompt**: D^* mesons produced directly at the primary vertex (PV), typically from charm quark hadronization.
- **Non-prompt**: D^* mesons originating from decays of long-lived beauty hadrons ($B \to \mathrm{D}^* + X$), leading to displaced decay topologies.
- **Background**: Combinatorial candidates formed from random track combinations that accidentally satisfy the D^* reconstruction criteria.

Separate ROOT files are used for each class, providing truth-labeled samples for supervised machine learning.

12.3 Transverse Momentum Selection

All studies in this section are performed within a fixed transverse momentum interval:

$$6 < p_{\mathrm{T}} < 8 \text{ GeV}/c,$$

where the transverse momentum of the reconstructed D^* candidate is defined as

$$p_{\mathrm{T}} = \sqrt{p_x^2 + p_y^2}.$$

The use of transverse momentum is motivated by the cylindrical geometry of collider detectors and the fact that particle production rates and decay topologies depend strongly on p_{T}.

12.4 Normalization of Distributions

All histograms are shown as *normalized candidate densities.* Each distribution is scaled such that

$$\int \frac{1}{N}\frac{dN}{dx}\,dx = 1,$$

allowing direct comparison of shape differences between prompt, non-prompt, and background candidates independent of their absolute yields.

12.5 Primary and Secondary Vertices: Physical Interpretation

The physically meaningful displacement information in this analysis originates exclusively from the decay of the D^0 meson. The decay

$$\mathrm{D}^* \to \mathrm{D}^0\pi$$

is a strong decay with an extremely small decay length and therefore provides no usable spatial separation from the primary vertex.

Consequently:

- The secondary vertex (SV) of the D^* is not used.
- All decay length and impact parameter variables are computed using the D^0 decay vertex.

This distinction is essential for the correct physical interpretation of the topological observables.

12.6 Topological Variables Used for Training

12.6.1 Cosine of the Pointing Angle (`fCpaD0`)

The cosine of the pointing angle is defined as

$$\cos\theta = \frac{\vec{L}\cdot\vec{p}}{|\vec{L}||\vec{p}|},$$

where $\vec{L}$ is the displacement vector from the primary vertex to the D^0 decay vertex, and $\vec{p}$ is the reconstructed D^0 momentum vector.

For true prompt decays, the momentum vector is expected to point back to the primary vertex, resulting in values of $\cos\theta$ close to unity.

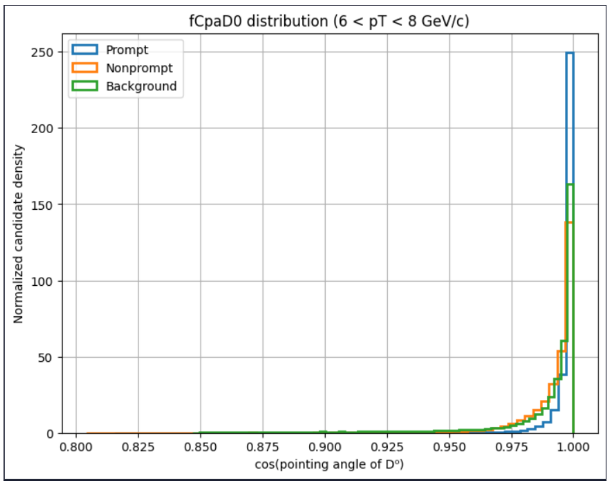


Figure 3: Distribution of `fCpaD0` for prompt, non-prompt, and background candidates in the interval $6 < p_T < 8$ GeV/c.

12.6.2 Cosine of the Pointing Angle in the Transverse Plane (`fCpaXYD0`)

This variable is defined analogously to `fCpaD0`, but using only the transverse (x–y) components of the displacement and momentum vectors. Due to detector resolution effects and the dominance of transverse geometry, this variable is strongly correlated with `fCpaD0`.

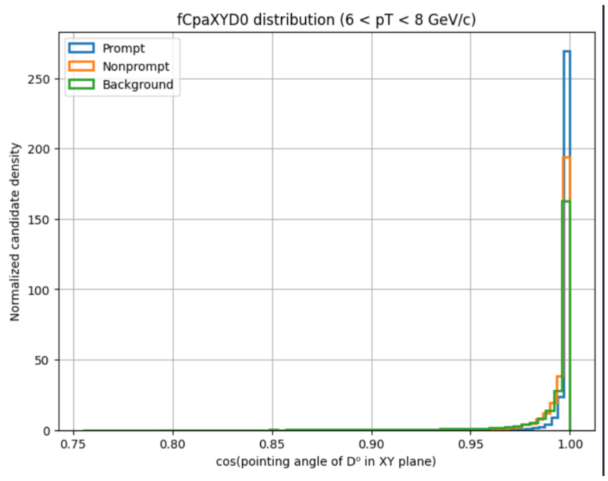


Figure 4: Distribution of `fCpaXYD0` for prompt, non-prompt, and background candidates.

12.6.3 Decay Length in the Transverse Plane (`fDecayLengthXYD0`)

The transverse decay length is defined as the magnitude of the displacement between the primary vertex and the D^0 decay vertex in the transverse plane:

$$L_{xy} = |\vec{r}_{SV} - \vec{r}_{PV}|_{xy}.$$

Prompt candidates are expected to peak near zero, while non-prompt candidates exhibit a tail towards larger values due to the finite lifetime of beauty hadrons.

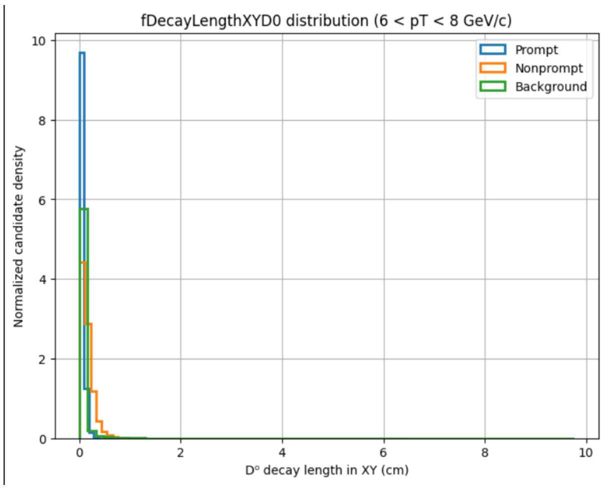


Figure 5: Distribution of `fDecayLengthXYD0` showing a clear separation between prompt and non-prompt candidates

12.6.4 Impact Parameter Product (`fImpactParameterProductD0`)

This variable is defined as the product of the impact parameters of the two D^0 daughter tracks with respect to the primary vertex. Large values indicate tracks inconsistent with originating from the PV, a characteristic feature of displaced decays.

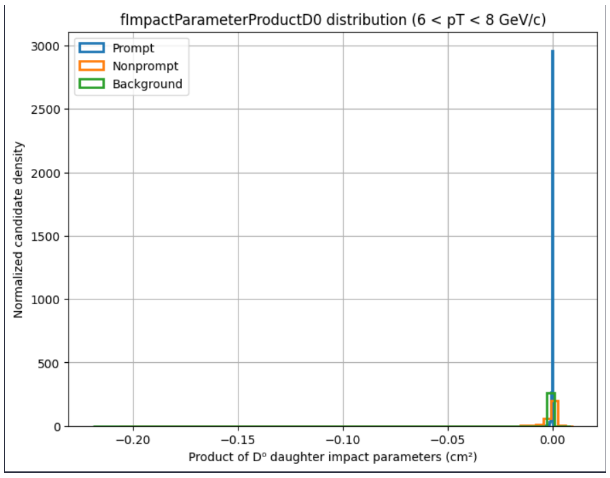


Figure 6: Distribution of `fImpactParameterProductD0` for the three candidate classes

12.6.5 Maximum Normalized Impact Parameter Difference (`fMaxNormalisedDeltaIPD0`)

The normalized impact parameter difference is defined as

$$\Delta IP/\sigma_{IP},$$

where ΔIP is the difference between the measured impact parameter and the expected value for a prompt decay, and σ_{IP} is the corresponding uncertainty.

The maximum value among the daughter tracks is used to enhance sensitivity to displaced decays.

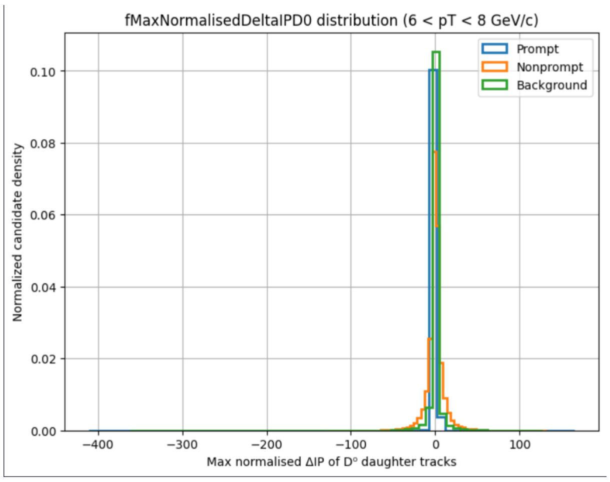


Figure 7: Distribution of `fMaxNormalisedDeltaIPD0`, illustrating broader tails for background candidates

12.6.6 Impact Parameter of the Soft Pion (`fImpParamSoftPi`)

In the decay channel

$$D^{*+} \to D^0 \pi^+_{\text{soft}},$$

the so-called *soft pion* carries a very small momentum due to the small mass difference between the D^* and D^0. Although the decay itself is prompt and occurs essentially at the D^* production point, the impact parameter (IP) of the soft pion with respect to the primary vertex provides additional discriminating power.

The impact parameter of the soft pion is defined as the distance of closest approach of the pion track to the primary vertex in the transverse plane. While the D^* does not have a measurable decay length, the origin of the soft pion track depends on the production mechanism of the parent hadron:

- For **prompt** D^* production, the soft pion originates very close to the primary vertex, leading to small impact parameter values.
- For **non-prompt** D^* mesons produced from B-hadron decays, the entire decay chain is displaced, causing the soft pion track to exhibit a larger impact parameter.
- **Background** candidates show a broad distribution due to random track combinations.

Therefore, although the soft pion does not define a secondary vertex, its impact parameter serves as an indirect probe of the displaced production topology and complements the D^0-based variables.

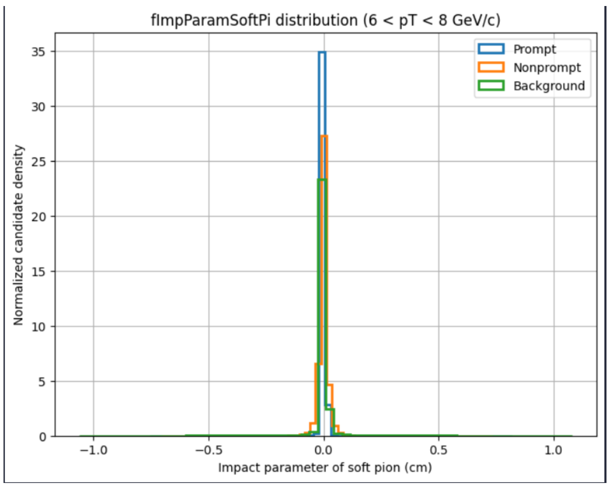


Figure 8: Distribution of the soft pion impact parameter (`fImpParamSoftPi`) for prompt, non-prompt, and background candidates in the interval $6 < p_{\mathrm{T}} < 8$ GeV/c

12.7 Kinematic Variables (Not Used for Training)

In addition to topological variables, several kinematic observables are examined for validation purposes but excluded from training to avoid bias.

[H]

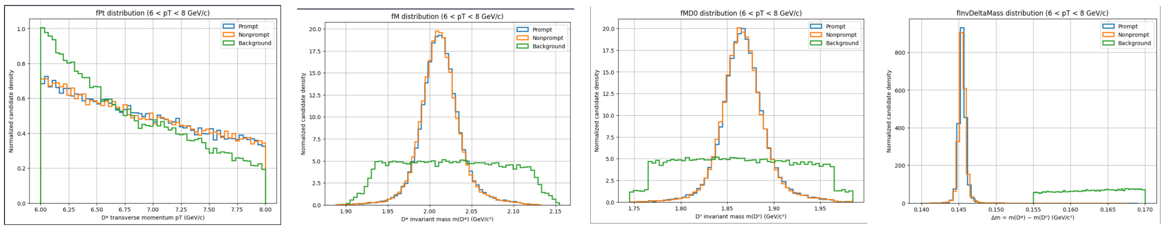


Figure 9: Distributions of kinematic variables (`fPt`, `fM`, `fMD0`, `fInvDeltaMass`) used for validation but excluded from training.

12.8 Summary of Exploratory Analysis

The exploratory analysis confirms that the selected topological variables exhibit strong and physically motivated separation between prompt, non-prompt, and background candidates. In particular, variables sensitive to decay displacement and pointing geometry provide the dominant discrimination power, justifying their use as inputs to the multivariate classifier.

13 Correlation Analysis of Input Variables

Before proceeding to multivariate training, it is essential to study the correlations among the selected input variables. A multivariate classifier, such as a Boosted Decision Tree (BDT), benefits most when its input features provide complementary and independent information. Highly correlated variables can introduce redundancy, reduce interpretability, and in some cases lead to overtraining.

13.1 Statistical Definition of Correlation

The linear correlation between two random variables x and y is quantified using the Pearson correlation coefficient:

$$\rho_{xy} = \frac{\mathrm{cov}(x,y)}{\sigma_x \sigma_y},$$

where the covariance is defined as

$$\mathrm{cov}(x,y) = \langle (x - \langle x \rangle)(y - \langle y \rangle) \rangle,$$

and σ_x, σ_y are the standard deviations of x and y, respectively. The Pearson coefficient satisfies

$$-1 \le \rho_{xy} \le 1,$$

with $\rho_{xy} = 0$ indicating no linear correlation, and $|\rho_{xy}| = 1$ indicating perfect linear correlation.

In this analysis, correlations are evaluated using reconstructed D* candidates in the transverse momentum interval

$$6 < p_{\mathrm{T}} < 8 \text{ GeV}/c.$$

13.2 Correlation Matrix Construction

A correlation matrix is constructed using the selected topological variables:

- Cosine of the pointing angle: `fCpaD0`
- Cosine of the pointing angle in the transverse plane: `fCpaXYD0`
- Transverse decay length: `fDecayLengthXYD0`
- Impact parameter product: `fImpactParameterProductD0`
- Soft-pion impact parameter: `fImpParamSoftPi`
- Maximum normalized impact parameter difference: `fMaxNormalisedDeltaIPD0`

Each matrix element represents the Pearson correlation coefficient between a pair of variables, computed over all selected candidates.

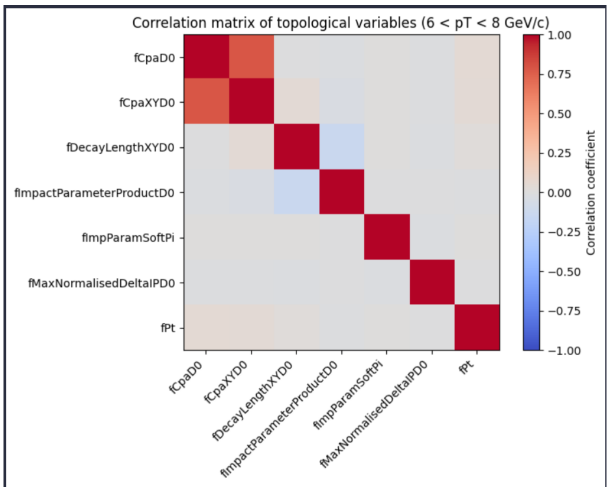


Figure 10: orrelation matrix of the topological variables used as inputs to the multivariate classifier, evaluated in the interval $6 < p_{\mathrm{T}} < 8$ GeV/c.

13.3 Physical Origin of Observed Correlations

The strongest correlation is observed between `fCpaD0` and `fCpaXYD0`. This behavior is expected from a geometrical standpoint. Both variables measure the alignment between the reconstructed D^0 momentum vector $\vec{p}_{\mathrm{D}^0}$ and the displacement vector $\vec{L}$ connecting the primary vertex (PV) to the D^0 decay vertex (SV).

The three-dimensional pointing angle is defined as:

$$\cos\theta_{3\mathrm{D}} = \frac{\vec{p}_{\mathrm{D}^0} \cdot \vec{L}}{|\vec{p}_{\mathrm{D}^0}||\vec{L}|},$$

while the transverse pointing angle is obtained by projecting both vectors onto the transverse plane:

$$\cos\theta_{\mathrm{XY}} = \frac{\vec{p}_{\mathrm{T}} \cdot \vec{L}_{\mathrm{XY}}}{|\vec{p}_{\mathrm{T}}||\vec{L}_{\mathrm{XY}}|}.$$

Since both quantities encode the same underlying physical concept—namely the consistency of the reconstructed momentum direction with the decay topology—their strong correlation is expected and physically well understood.

13.4 Independence of Other Topological Variables

Other variables, such as the transverse decay length, impact parameter product, and normalized impact parameter differences, exhibit weaker correlations. This indicates that these variables probe different aspects of the decay topology:

- Decay length variables capture the spatial displacement of the D^0 decay vertex.
- Impact parameter variables quantify the deviation of daughter tracks from the primary vertex.
- Soft-pion impact parameter provides indirect sensitivity to the production mechanism of the parent hadron.

The relatively low correlations among these variables imply that they provide complementary information, allowing the multivariate classifier to learn non-trivial decision boundaries.

13.5 Implications for Multivariate Training

While strong correlations can, in principle, be reduced by removing redundant variables, both `fCpaD0` and `fCpaXYD0` are retained in the present analysis. Boosted Decision Trees are known to be robust against moderate correlations, and retaining both variables allows the classifier to exploit differences arising from detector resolution and projection effects.

This correlation study validates the choice of input variables and provides confidence that the classifier training will be driven by genuine physical differences between prompt, non-prompt, and background candidates rather than statistical artifacts.

14 Multivariate Classification Performance and Working Point Selection

After validating the physical relevance and mutual correlations of the selected topological variables, a multivariate classifier was trained to separate prompt D^* candidates from non-prompt and background contributions. In this section, the performance of the trained model is evaluated using Receiver Operating Characteristic (ROC) curves, and a physics-motivated working point is selected.

14.1 Classifier Output and Interpretation

The Boosted Decision Tree (BDT) is trained in a multi-class configuration, producing for each reconstructed D* candidate a set of class probabilities:

$$P(\text{background}), \quad P(\text{non-prompt}), \quad P(\text{prompt}),$$

with the constraint that the probabilities sum to unity. In the following, the classifier output

$$s \equiv P(\text{prompt})$$

is used as the discriminating variable.

Rather than assigning a hard class label, the continuous score $s \in [0, 1]$ is retained. This is essential in physics analyses, where the final selection must be tunable depending on the desired balance between signal efficiency and background rejection.

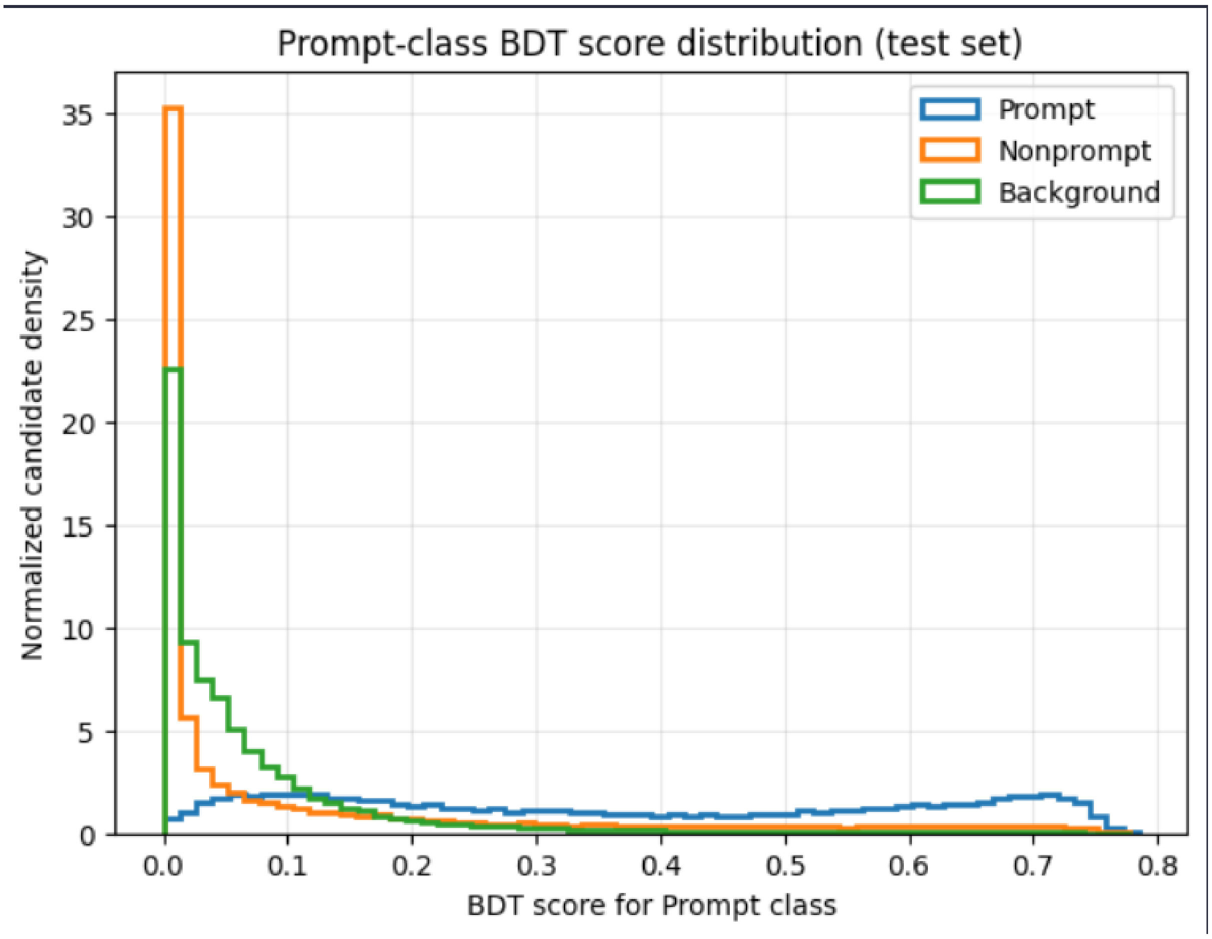


Figure 11: BDT score distribution

14.2 Receiver Operating Characteristic (ROC) Curves

To evaluate the separation power of the classifier independently of any fixed threshold, Receiver Operating Characteristic (ROC) curves are constructed.

For a given threshold s_{cut}, two quantities are defined:

- **True Positive Rate (TPR)**, or prompt efficiency:

$$\text{TPR}(s_{\text{cut}}) = \frac{N_{\text{prompt}}(s > s_{\text{cut}})}{N_{\text{prompt}}^{\text{total}}}.$$

- **False Positive Rate (FPR)**, or background efficiency:

$$\mathrm{FPR}(s_{\mathrm{cut}}) = \frac{N_{\mathrm{background}}(s > s_{\mathrm{cut}})}{N^{\mathrm{total}}_{\mathrm{background}}}.$$

An ROC curve is obtained by scanning s_{cut} from 0 to 1 and plotting TPR as a function of FPR.

14.2.1 Prompt vs Background Separation

The first ROC curve evaluates the ability of the classifier to separate prompt D* candidates from combinatorial background. This represents the most critical task, as background candidates arise from random track combinations with poorly defined decay topology.

The area under the ROC curve (AUC) is found to be:

$$\mathrm{AUC}_{\mathrm{prompt–background}} = 0.961.$$

This value has a clear probabilistic interpretation: if one prompt candidate and one background candidate are chosen at random, there is a 96.1% probability that the classifier assigns a higher prompt score to the prompt candidate. This demonstrates strong separation power driven by topological differences.

14.2.2 Prompt vs Non-prompt Separation

A second ROC curve is constructed to evaluate the separation between prompt and non-prompt D* candidates. This is a more challenging problem, as both classes correspond to genuine charm mesons and differ primarily in their displacement from the primary vertex.

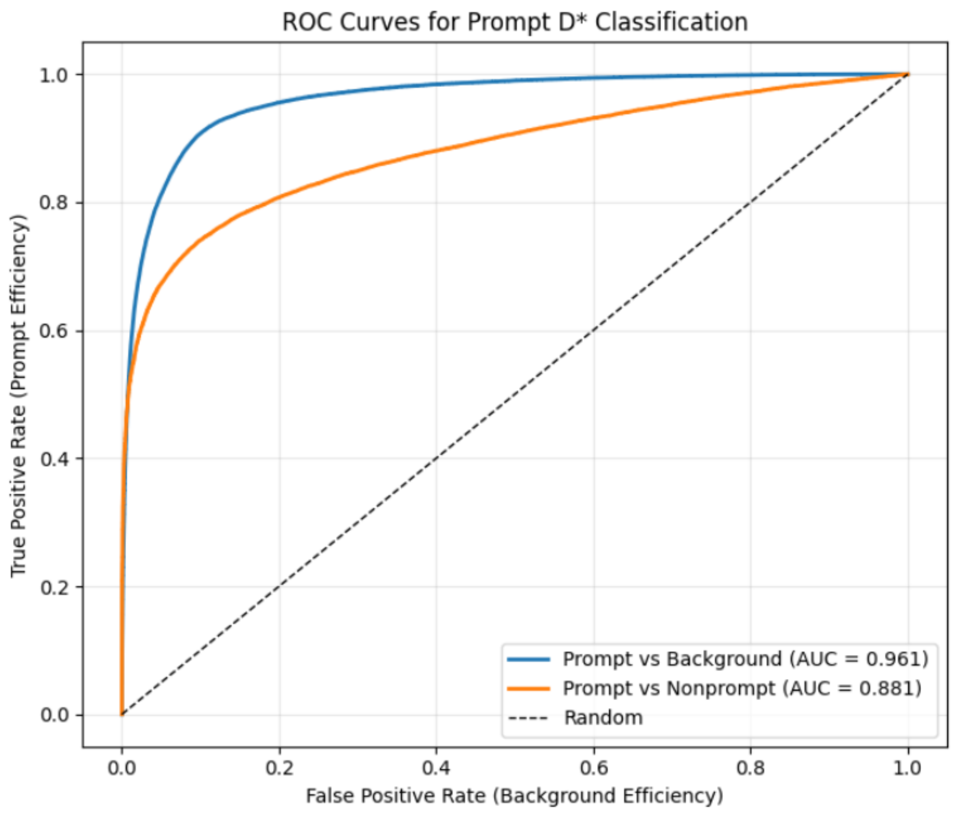


Figure 12: ROC curve for prompt versus non-prompt D* candidates in the interval $6 < p_{\mathrm{T}} < 8$ GeV/c

The resulting area under the curve is:

$$\mathrm{AUC}_{\mathrm{prompt–nonprompt}} = 0.881.$$

This lower AUC, compared to the prompt–background case, reflects the intrinsic physical similarity between prompt and non-prompt charm decays. Nonetheless, the classifier provides substantial discrimination based on decay length and impact parameter information.

14.3 Choice of a Physics-Motivated Working Point

While ROC curves provide a global measure of classifier performance, a concrete physics analysis requires the selection of a specific operating point, corresponding to a fixed threshold on the classifier output.

In this analysis, a **high-efficiency working point** is chosen, targeting a prompt efficiency of approximately 85%. This choice is motivated by the desire to retain the majority of prompt D^* candidates while achieving significant background suppression, which is particularly suitable for subsequent studies of yields and spectra.

The working point is defined by selecting the threshold s_{cut} such that:

$$\mathrm{TPR}(s_{\mathrm{cut}}) \approx 0.85$$

on the prompt versus background ROC curve.

The corresponding classifier threshold is found to be:

$$s_{\mathrm{cut}} = 0.1816.$$

14.4 Efficiencies at the Selected Working Point

Applying this threshold to the independent test dataset yields the following efficiencies:

- **Prompt efficiency:**
$$\epsilon_{\mathrm{prompt}} = 0.850,$$
indicating that 85% of true prompt D^* candidates are retained.

- **Background efficiency:**
$$\epsilon_{\mathrm{background}} = 0.065,$$
corresponding to a rejection of approximately 93.5% of combinatorial background.

- **Non-prompt efficiency:**
$$\epsilon_{\mathrm{nonprompt}} = 0.303,$$
meaning that roughly 30% of non-prompt D^* candidates pass the prompt selection.

The non-prompt efficiency quantifies the level of beauty feed-down contamination remaining after the selection. This behavior is expected, as non-prompt D^* mesons are physically similar to prompt ones and differ primarily through displacement-related observables.

14.5 Summary of Classifier Performance

The trained multivariate classifier demonstrates strong and physically interpretable performance. At a working point retaining 85% of prompt candidates, the model achieves substantial background rejection while moderately suppressing non-prompt contributions. This balance is well suited for further physics studies and mirrors strategies commonly adopted in experimental charm analyses at the LHC.

The classifier output and selected working point will be used in subsequent steps to test performance on mixed datasets and to evaluate the impact of additional particle identification variables.

15 Incorporation of Particle Identification Variables

15.1 Motivation

In the baseline analysis, the machine learning classifier is trained using only topological variables describing the decay geometry of reconstructed D-meson candidates. While these observables are effective for separating prompt and non-prompt decays, they do not provide information about the particle species of the daughter tracks.

During reconstruction, particle hypotheses (kaon or pion) are assigned to tracks based on kinematic assumptions. Incorrect assignments lead to misidentified decay candidates and contribute to combinatorial background.

Particle Identification (PID) variables quantify the compatibility of each track with different particle species hypotheses. By including PID information as additional input features, the classifier gains access to information about both decay geometry and particle identity.

The extended feature set is therefore constructed as

$$\vec{x} = (\vec{x}_{\text{topo}}, \vec{x}_{\text{PID}}),$$

where $\vec{x}_{\text{topo}}$ denotes the original topological variables and $\vec{x}_{\text{PID}}$ denotes the PID variables.

All other aspects of the machine learning pipeline remain unchanged.

15.2 Definition of Particle Identification Variables

Particle Identification (PID) is based on comparing measured detector responses from charged-particle tracks with the expected responses for different particle species hypotheses, such as pions, kaons, and protons. Two detector subsystems are primarily used: the Time Projection Chamber (TPC), which measures the specific energy loss dE/dx, and the Time-Of-Flight (TOF) detector, which measures the particle flight time.

For a given detector observable S and a particle hypothesis h, the deviation between the measured signal and the expected signal is expressed in units of the detector resolution. This quantity is defined as

$$n\sigma_h = \frac{S_{\text{measured}} - S^{(h)}_{\text{expected}}}{\sigma_{\text{detector}}}.$$

Here, S_{measured} is the measured detector response, $S^{(h)}_{\text{expected}}$ is the expected response assuming particle hypothesis h, and σ_{detector} is the corresponding detector resolution.

The definition is applied independently for the TPC and TOF detectors, leading to PID variables such as $n\sigma_\pi^{\text{TPC}}$, $n\sigma_K^{\text{TPC}}$, $n\sigma_\pi^{\text{TOF}}$, and $n\sigma_K^{\text{TOF}}$.

Small absolute values of $n\sigma_h$ indicate high compatibility of the track with the particle hypothesis h, while large absolute values indicate low compatibility.

In this analysis, the raw $n\sigma$ values are used directly as input features to the neural network, allowing the model to learn optimal combinations of PID and topological information.

15.3 Interpretation and Selection of PID Variables

The reconstructed decay channel considered in this analysis is

$$D^0 \to K^- + \pi^+.$$

This decay produces two charged daughter tracks. In the data format, these daughter tracks are stored using an index notation:

- Prong 0: first daughter track,
- Prong 1: second daughter track.

The prong index is purely an array index and does not encode any physical particle type.

15.3.1 PID Variable Naming Convention

Particle Identification (PID) variables are stored using the following pattern:

`fNSig[Detector][Hypothesis][ProngIndex]`.

For example:

- `fNSigTpcPi0`: nσ value in TPC assuming pion hypothesis for prong 0,
- `fNSigTpcKa1`: nσ value in TPC assuming kaon hypothesis for prong 1,
- `fNSigTofPi0`: nσ value in TOF assuming pion hypothesis for prong 0.

The symbols `Pi` and `Ka` denote the *particle hypothesis* used when computing the expected detector response. They do not represent the true particle type of the track. Each physical track is tested under multiple hypotheses.

Thus, a single track simultaneously has:

$$n\sigma_\pi, \quad n\sigma_K$$

values, corresponding to its compatibility with pion and kaon hypotheses.

15.3.2 Interpretation

Small absolute values of $n\sigma$ indicate high compatibility with the corresponding hypothesis, while large values indicate poor compatibility. For example,

$$|n\sigma_K| \ll |n\sigma_\pi|$$

implies that the track is more kaon-like than pion-like.

The presence of both pion and kaon hypotheses for the same track allows the machine learning model to compare hypotheses rather than relying on a single assumption.

15.3.3 Selected PID Variables

Since the signal decay channel contains only pions and kaons, only pion and kaon hypotheses are used. Proton hypotheses are not included in order to limit the dimensionality of the feature space.

For each prong, the following PID variables are selected:

- TPC nσ for pion hypothesis,
- TPC nσ for kaon hypothesis,
- TOF nσ for pion hypothesis,
- TOF nσ for kaon hypothesis.

This results in four PID variables per prong.

15.3.4 Final PID Feature List

For prong 0:

- `fNSigTpcPi0`,
- `fNSigTpcKa0`,
- `fNSigTofPi0`,
- `fNSigTofKa0`.

For prong 1:

- `fNSigTpcPi1`,
- `fNSigTpcKa1`,
- `fNSigTofPi1`,
- `fNSigTofKa1`.

The total number of PID variables used in this analysis is therefore

$$N_{\text{PID}} = 8.$$

When combined with the six baseline topological variables, the final input feature vector has

$$N_{\text{features}} = 6 + 8 = 14$$

components.

15.4 Construction of Feature Matrix and Class Labels

After extracting the selected topological and PID variables from the ROOT files, the next step is to construct the input feature matrix for machine learning.

For each dataset (prompt, non-prompt, and background), the variables are stacked column-wise to form a feature matrix,

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1,14} \\ x_{21} & x_{22} & \cdots & x_{2,14} \\ \vdots & \vdots & & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{N,14} \end{bmatrix},$$

where each row corresponds to one D-meson candidate and each column corresponds to one input feature.

Class labels are assigned as follows:

- Prompt candidates: $y = 0$,
- Non-prompt candidates: $y = 1$,
- Background candidates: $y = 2$.

The feature matrices from the three datasets are concatenated to form a single input matrix,

$$X = \begin{bmatrix} X_{\text{prompt}} \\ X_{\text{nonprompt}} \\ X_{\text{bkg}} \end{bmatrix},$$

and the corresponding label vectors are concatenated to form

$$y = (y_{\text{prompt}}, y_{\text{nonprompt}}, y_{\text{bkg}}).$$

The resulting dataset (X, y) is used as input to the neural network.

15.5 Feature Normalization

The input features exhibit different numerical scales. In order to ensure stable and efficient neural network training, each feature is normalized to zero mean and unit variance.

For a given feature x, the normalized value $\tilde{x}$ is defined as

$$\tilde{x} = \frac{x - \mu}{\sigma},$$

where μ and σ denote the mean and standard deviation of the feature, respectively.

The dataset is first split into training and test samples. The normalization parameters μ and σ are computed using only the training sample in order to avoid information leakage. The same transformation is then applied to both the training and test sets.

This procedure ensures that all input features contribute comparably to the optimization process.

15.6 Neural Network Architecture

A fully connected feed-forward neural network is used for the classification task. The network consists of three hidden layers followed by an output layer with three neurons corresponding to the prompt, non-prompt, and background classes.

The architecture is defined as

$$14 \to 64 \to 32 \to 16 \to 3,$$

where the input dimension corresponds to the 14 input features.

Rectified Linear Unit (ReLU) activation functions are applied after each hidden layer. The output layer produces raw scores (logits). A softmax operation is applied internally within the loss function during training.

This architecture provides sufficient capacity to model nonlinear correlations among the input features while avoiding excessive model complexity.

15.6.1 Loss Function and Optimization

For the three-class classification problem, the categorical cross-entropy loss function is used. This loss measures the difference between the predicted class probabilities and the true class labels.

The network parameters are optimized using the Adam optimizer with an initial learning rate of 10^{-3}. Adam provides adaptive learning rates for individual parameters and ensures stable convergence.

In order to improve convergence and fine-tuning, a learning-rate scheduler with step decay is employed. The learning rate is reduced by a factor of 0.5 every 20 training epochs. This allows large parameter updates during early training and smaller updates during later stages.

15.6.2 Training Procedure

The neural network is trained using a standard supervised learning procedure. The normalized training features are provided as input to the model, which produces output scores for the three classes.

For each training epoch, a forward pass is performed to compute the network outputs. The cross-entropy loss between the predicted outputs and the true labels is then evaluated. Gradients of the loss with respect to the network parameters are computed using backpropagation. The parameters are updated using the Adam optimizer.

After each epoch, the learning-rate scheduler updates the learning rate according to the step-decay policy. The training loss value is recorded at every epoch in order to monitor convergence.

The training procedure is repeated for a fixed number of epochs.

16 Training Performance

The neural network is trained for a fixed number of epochs using mini-batch gradient descent with the Adam optimizer. During training, the value of the cross-entropy loss is recorded at each epoch in order to monitor convergence.

Figure **??** shows the training loss as a function of epoch. A monotonic decrease of the loss indicates stable learning and convergence of the network.

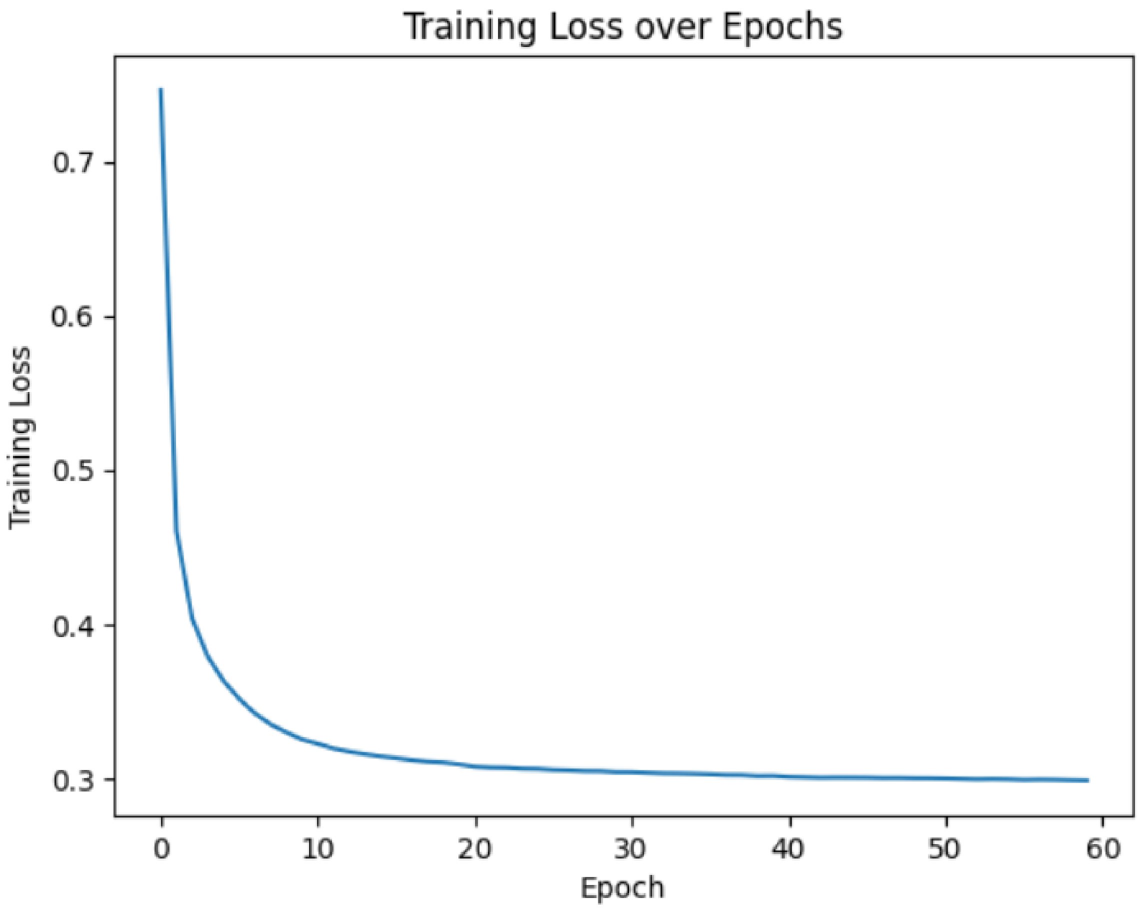


Figure 13: Training loss as a function of epoch for the neural network

Classification Performance

The trained neural network is evaluated using the independent test dataset. The predicted class for each event is obtained by selecting the output node with the maximum predicted probability.

The trained neural network achieves an overall classification accuracy of

$$\text{Accuracy} = 0.885.$$

The corresponding confusion matrix obtained using the independent test dataset is shown in Table 1. Rows correspond to the true class, while columns correspond to the predicted class.

True \ Predicted	Prompt	Non-prompt	Background
Prompt	19825	2839	6021
Non-prompt	6078	26613	3054
Background	4107	2979	146649

Table 1: Confusion matrix for the neural network classifier evaluated on the test dataset.

The diagonal elements of the confusion matrix indicate that a large fraction of prompt, non-prompt, and background candidates are correctly classified. Misclassifications primarily arise from confusion between prompt and non-prompt candidates, which is expected due to their similar decay topologies. Background candidates are classified with high efficiency, indicating strong background rejection.

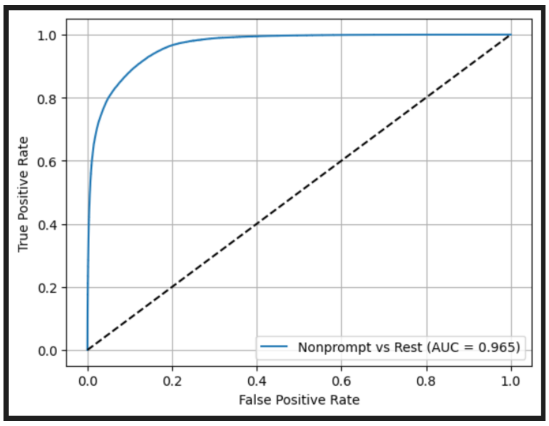
Receiver Operating Characteristic Curves

To quantify the discrimination power of the neural network, receiver operating characteristic (ROC) curves are computed using a one-vs-rest strategy.

ROC curves are produced for:

- Prompt versus (non-prompt + background),
- Non-prompt versus (prompt + background).

The area under each ROC curve (AUC) is used as a summary measure of performance. The ROC curves are shown in Figure.



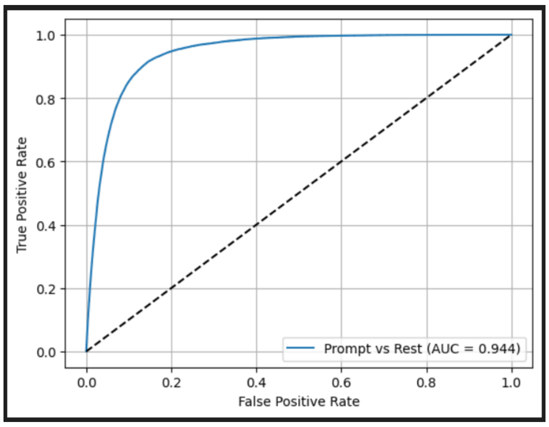


Figure 14: ROC curves for prompt and non-prompt classification using the neural network.

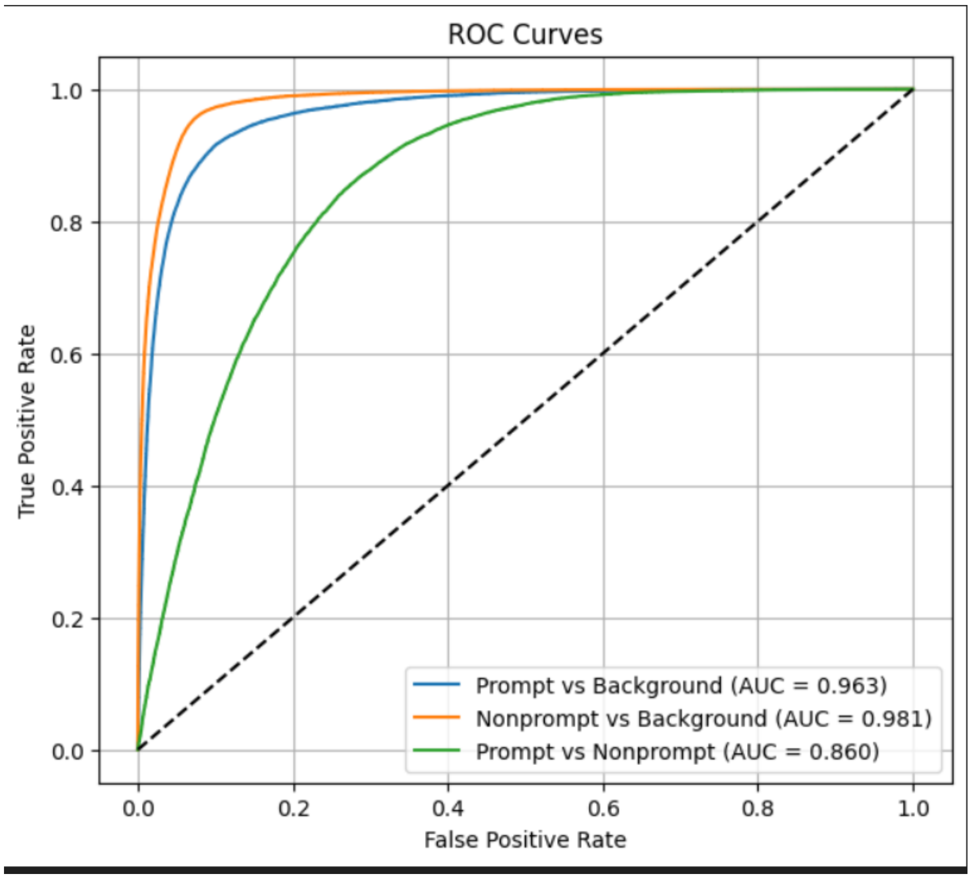


Figure 15: ROC Curve